A 2D Graphic Editor Using MAGOO,

An Object Oriented Graphic Library.

David Raynes (dwr@minerva.inesc.pt)

Fernando Vasconcelos (fmhv@minerva.inesc.pt)

INESC

Instituto de Engenharia de Sistemas e Computadores

Rua Alves Redol No 9, sala 208

Apartado 10105 1017 Lisboa CODEX - Portugal

Tel: +351(1)545150 Ext. 216

Fax: +351(1)525843

Abstract

The main object of the application described in this article was to demonstrate the ease of use and functionality of the Magoo architecture. This exercise has shown us some of the problems still unsolved and helped us to improve both the Magoo interface and functionality.

1 Introduction

The application program outlined in this article facilitates the interactive creation and manipulation of two-dimensional graphic objects, through means of the MAGOO architecture.

The MAGOO architecture provides a comparatively simplified means of generating a high level user interface for graphic application programs. MAGOO also provides a means of abstraction between virtual graphic objects and the application programs which wish to manipulate them.

One of the primary aims in the construction of the editor, was to provide a complete, yet 'fool-proof' user interface which, simplifies the user operations while, at the same time, increases the user's control over the virtual objects he wishes to create.

Visually the editor consists of a main menu bar, a graphics drawing area, an icon box and a palette status box. The icon box contains an array of creation mode icons together with the selection mode icon. Each icon corresponds to the respective editor mode.

2 Program Functionality

2.1 Object Creation

The editor facilitates the interactive creation of Primitive Graphic Objects (PGOs). The actual set of PGO classes provided by the editor include Vertical/Horizontal Line, Any Line, Rectangle, Polygon, Circle, Ellipse and Spline .

Each PGO class has an associated creation mode, which is set by the user by clicking the mouse over the respective creation mode icon. Each creation mode is implemented as a dialog, which requires the interactive user input of the various specification points required to create the instance of the respective PGO class. Only the minimum number of specification points are required, eg. for a rectangle, only two points are required as input.

After the user has specified the first specification point, a temporary object is displayed, which allows the object to be visualized during the creation process.

2.2 Object Selection

A single PGO or collection of PGOs can be selected by the user, forming the Subject. The Subject is defined as a set containing one or more PGOs. This set can be treated as a single entity by each on of the Collective Modification Operations (CMOs).

Selection is achieved by first entering Selection Mode, ie. clicking the mouse over the Selection Mode Icon. The selection process can then be divided into Single Selection and Collective Selection.

Single Selection permits the selection of a single PGO and is achieved by clicking the mouse over (or near) one of the specification points of the required PGO. When this is done the Subject will contain the required PGO only.

Collective Selection permits the selection of one or more PGOs and is achieved by dragging a temporary frame over the required PGO's. When this is done all PGOs lying inside the temporary frame will be selected and thus become member PGOs of the Subject.

When the subject has been selected, whether it be through Single Selection or Collective Selection, the Specification Points of each member PGO of the Subject will be highlighted. At this point any one of the modification operations described below can be applied to the Subject.

2.3 Object Modification

The set of modification operations can be partitioned into two groups, Individual Modification Operations and Collective Modification Operations.

2.3.1 Direct Manipulation

Direct Manipulation is an Individual Modification Operation, therefore it is not possible to directly manipulate more than one PGO at any one instance in time.

By clicking on one of the highlighted specification points of a selected PGO, while in selection mode, that point of the PGO can be dragged with the mouse.

The exact effect of direct manipulation is dependent (or should be ...) upon the class of the PGO being modified. This is because a PGO must always conform to it's class specification, ie. an instance of the PGO class rectangle must always consist of exactly four sides where adjacent sides are at right-angles to each other.

2.3.2 Collective Modification Operations

As far as collective modification operations are concerned the editor provides three sorts of transformations which are translations, rotations and scaling.

For each transformation operation the user can specify the exact transformation by manipulating a temporary frame, which is drawn around the Subject.

2.4 Object Duplication

Duplication creates an exact duplicate of the Subject. Once this is done, all the duplicate PGOs become the Subject and the editor is left in transformation mode, allowing the duplicate Subject to be moved to the required position.

2.5 Object Deletion and Clearing the Drawing Area

Deletion removes and deletes every member PGO of the subject, whereas Clear removes every PGO in existence, after first warning the user.

2.6 The Palette

When a PGO is created it takes on the current attribute and mode settings. These current attribute and mode settings make up the Palette. The sets of Palette attributes provided by the application are the Foreground Colour, Background Colour, Line Width and Fill Pattern. As far as the drawing modes are concerned they include No Fill, Fill, and Transparent Fill.

These settings are constantly displayed in the Palette Status Box. Each attribute has an associated status icon, and each mode has an associated toggle button. The attribute values or mode settings can be altered at any time.

In order to alter a specific Palette attribute, the user simply need click the mouse over the corresponding attribute status icon in the Palette Status Box. On doing this a Value Selection Box will appear, from which the required new Palette attribute value can be set.

Mode alteration is achieved by inverting the respective mode toggle button in the Palette Status Box.

The alteration of a pre-created PGO's attributes and modes is also facilitated through this process. Palette alteration only, is made if there are no PGOs selected.

2.7 View Control Operations

The user is given control over the magnification of the graphics drawing area. Thus the user can Zoom and UnZoom the graphics drawing area.

The editor does not put a limit on the number of consecutive zooms or unzooms, but in practice the editor will lose point precision at very high or low magnifications of the graphics drawing area.

When unzooming the last view before the zoom operation is restored.

2.8 Option Control Operations

This set of operations allows the user to alter various display features of the editor. These include such things as Scroll Bars in the graphics drawing area, the mouse pointer type and the display of a grid over the graphics drawing area.

As yet only the grid control operations have been fully implemented .

2.9 File Operations

The editor should also provide a set of file operations. These are not implemented yet as MAGOO does not provide the basic functionality to save and restore objects. This is one of the planned extensions to the MAGOO architecture.

3 Implementation

This application program, like the MAGOO architecture, is implemented using the object-orientated programming language C++. The application runs on top of MAGOO on UNIX and X11.

The implementation of the application consists of four major parts, the User Interface Modules, the Main Interface Construction Module, a set of Callbacks and a set of Dialogs.

The User Interface Modules are an encapsulation of the MAGOO user interface toolkit part inside C++ classes. Each module normally consists of a collection of MAGOO widgets, which together achieve some aspect of the application user interface (in some cases only visual aspects).

The Main Interface Construction Module, constructs the user interface from the user interface modules, creates all required Dialogs other global objects and links the callbacks.

The Callbacks are a set of functions which together with the Dialogs, are responsible for all program functionality. The callbacks act as a link between the application user interface and the application functionality. Every main menu option, icon and push button has an associated callback function.

3.1 Dialogs

The Dialogs handle the more complex user interaction with the graphics drawing area, for example, the creation of an object or the specification of a transformation. Each Dialog is implemented as a C++ class and together with a constructor, has three defined member functions, LinkDisplay, UnlinkDisplay and ReceiveEvent.

Every dialog has a set of events which correspond to the mouse events (for instance button one down). These events can either be active or passive. On the occurrence of an active event control is passed to ReceiveEvent which acts on the event accordingly. The occurrence of a passive event is ignored.

When a dialog is created it is initially unlinked to the display, ie. it's events are all passive. The member LinkDisplay has the effect of initializing the events for the dialog.

For example, if the first expected event occurrence is the depression of the left-hand mouse button, then LinkDisplay will initialize the events of the dialog such that the event buttonOneDown will be active and all other events will be passive. The member UnlinkDisplay sets all events to passive.

ReceiveEvent is an implementation of a deterministic finite state automaton, that is, the dialog can only exist in one state at any one moment in time and for each specific state, on receiving a specific event only one possible state transition can occur.

Some dialogs control very similar processes, therefore the C++ inheritance facility has been put to use. The set of dialogs used in this application form an inheritance hierarchy as shown below.

```
                                     General
                                        |
                                        |
         +---------------+--------+-------+------------------+
         |               |                |                  |
         |               |                |                  |
     Selection        Creation           Zoom          Transformation
                         |                                   |
                         |                                   |
   +-----+-----+-----+-----+-----+-----+               +------+-----+
   |     |     |     |     |     |     |               |      |     |
   |     |     |     |     |     |     |               |      |     |
AnyLine  |   Circle  | Rectangle |   Spline   Translation  |   Rotation
         |           |           |                            |
         |           |           |                            |
 Vert/Horz Line   Ellipse      Polygon                      Scale
```

4 Documentation

Together with the implementation of the application, the complete program specification documentation and the user manual have also been produced.

5 Assessment

Construction of the user interface using the User Interface Modules together with a main construction module, works very well. This implementation allows a user interface to be modified or built from scratch very quickly. The coding of the User Interface Modules themselves is not particularly clever. This could have been done much better by making user of the inheritance capabilities provided by C++.

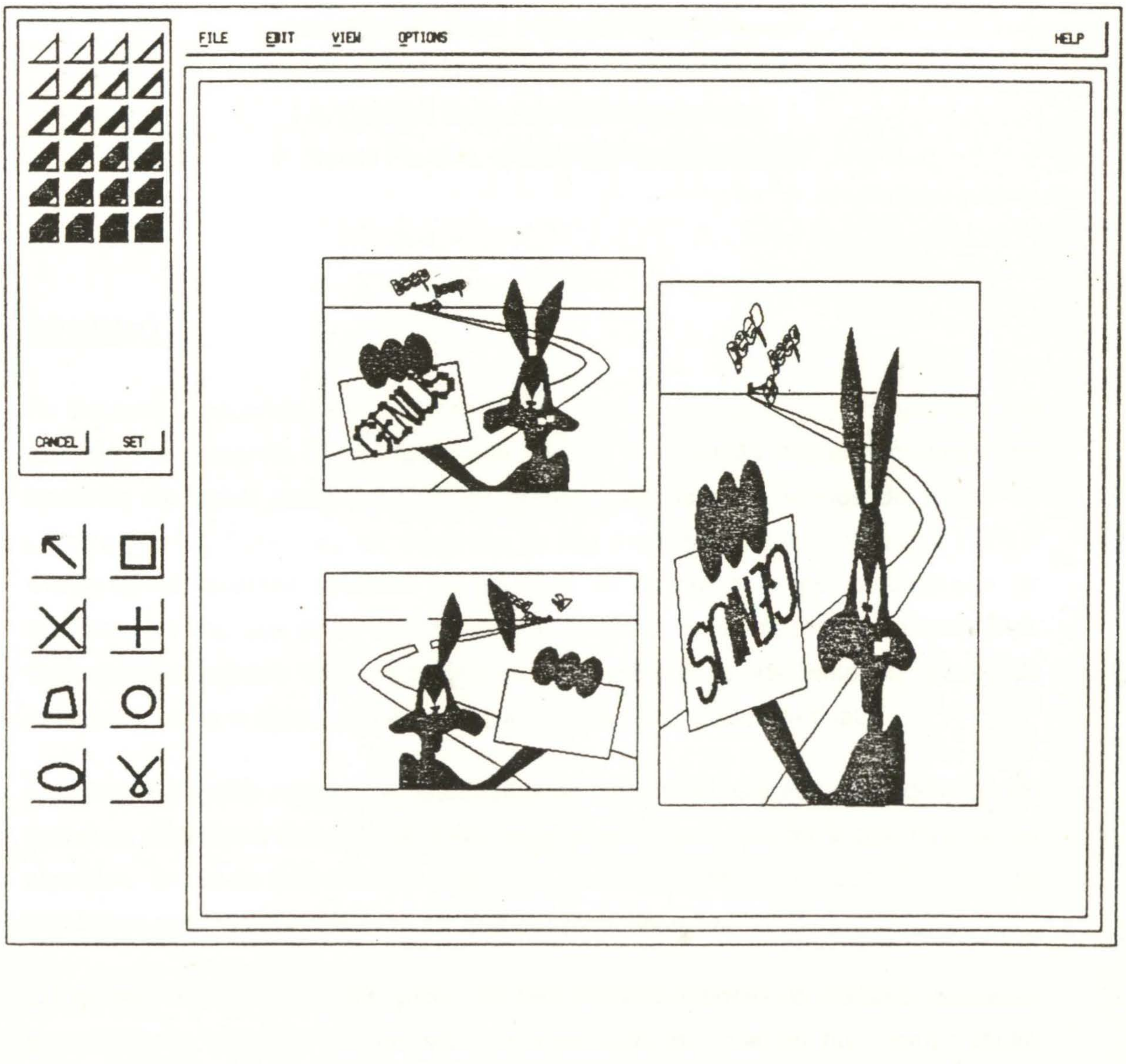
The implementation of the menu bar is especially successful as it permits the greatly simplified construction of arbitarily complex menus and sub-menus.

The use of callbacks clashes with object-orientated approach of C++, for this reason it would be preferable if user interaction was achieved solely though the use of dialogs. On the actual MAGOO version there is no longer need of callbacks, every user action may now be handled by a dialog.

With the primary aim of the project being that of the simplification of object creation, manipulation and modification, it is fair to say that this has been achieved successfully. These complicated processes of interaction are greatly simplified by the MAGOO provision for dialogs.

In the next page we present an example of a typical session with our 2D editor.

2D Graphics Editor


FILE
EDIT
VIEW
OPTIONS
HELP
CANCEL
SET
GENIUS
beep
beep